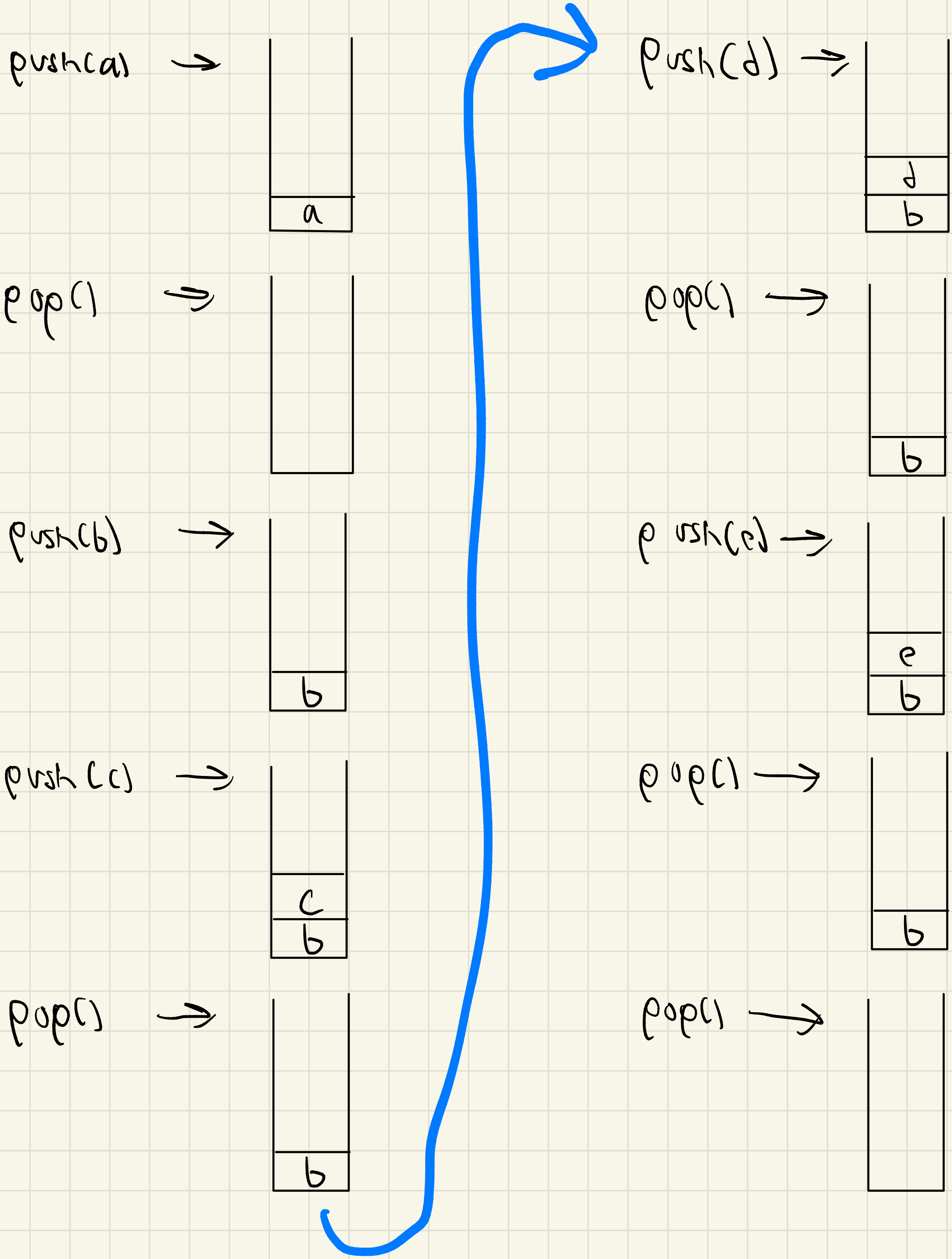
1.


push(a) →
a
pop() →
push(b) →
b
push(c) →
c
b
pop() →
b
push(d) →
d
b
pop() →
b
push(e) →
e
b
pop() →
b
pop() →

2. enqueue(a)

a

enqueue(b)

b
a

dequeue()

b

enqueue(c)

c
b

enqueue (d)

d
c
b

enqueue(e)

e
d
c
b

dequeue() →

e
d
c

dequeue() →

e
d

enqueue(f) →

f
e
d

4. For my solution, I started by creating an array as long as the number of lockers. I then iterated over this array with two for loops. The first for loop runs r times, where r is the number of passes. The second for loop runs for the length of the array. It takes the value at each index of the array and mods it by which pass the first for loop is on. If the result is 0, that means that locker number is a multiple of the pass number, and therefore has its open or closed value toggled. After both loops are complete, another for loop iterates through the completed array. This one prints the locker number and if it is open or not by checking if the value in the array is 1 or 0. If the locker is open, 1 is added to the numOpen variable, and it is printed at the end of the loop.

Problems: Initially, my main function called another function which would return an integer array. However, since the array was a local variable in the function, the values would be all wrong when it returned. I fixed this by making this function void and moving the printing from main to that function.

Compilation: Compile as normal. When running the program, pass n and r in as command line arguments. Ex. ./a.out 12 3

number of lockers

number of passes